Class Notes for CS 819 C - Spring 2002

3. Specifications and Inference Rules

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

revised February 6, 2002

Specifications

$$
\begin{array}{lll}
\langle\text{spec}\rangle ::= \{\langle\text{assert}\rangle\}\ \langle\text{comm}\rangle\ \{\langle\text{assert}\rangle\} & & \text{(partial correctness)} \\
\qquad\quad \mid [\ \langle\text{assert}\rangle\ ]\ \langle\text{comm}\rangle\ [\ \langle\text{assert}\rangle\ ] & & \text{(total correctness)}
\end{array}
$$

$$
[\![-]\!]_{\text{spec}} \in \langle\text{spec}\rangle \to \mathbf{B}.
$$

When $[\![sp]\!]_{\text{spec}}$ is true we say that *sp holds*.

Let $V = \text{FV}(p) \cup \text{FV}(c) \cup \text{FV}(q)$. Then

Partial correctness:

$\{p\}\ c\ \{q\}$ holds iff $\forall(s,h) \in \text{States}_V.\ [\![p]\!]_{\text{assert}} s\, h$ implies

$\neg\,(c, (s,h) \rightsquigarrow^* \mathbf{abort})$

and $(\forall(s',h') \in \text{States}_V.\ c, (s,h) \rightsquigarrow^* (s',h')$ implies $[\![q]\!]_{\text{assert}} s'\, h')$

Total correctness:

$[\ p\ ]\ c\ [\ q\ ]$ holds iff $\forall(s,h) \in \text{States}_V.\ [\![p]\!]_{\text{assert}} s\, h$ implies

$\neg\,(c, (s,h) \rightsquigarrow^* \mathbf{abort})$

and $\neg\,(c, (s,h) \uparrow)$

and $(\forall(s',h') \in \text{States}_V.\ c, (s,h) \rightsquigarrow^* (s',h')$ implies $[\![q]\!]_{\text{assert}} s'\, h')$.

- Specifications are implicitly quantified over both stores and heaps — and over all executions.

- An execution giving a memory fault falsifies a specification.

Examples of Partial Correctness Specifications

$$\{\mathsf{x}-\mathsf{y}>3\}\ \mathsf{x}:=\mathsf{x}-\mathsf{y}\ \{\mathsf{x}>3\}$$

$$\{\mathsf{x}+\mathsf{y}\geq 17\}\ \mathsf{x}:=\mathsf{x}+10\ \{\mathsf{x}+\mathsf{y}\geq 27\}$$

$$\{\mathsf{x}\leq 10\}\ \mathbf{while}\ \mathsf{x}\neq 10\ \mathbf{do}\ \mathsf{x}:=\mathsf{x}+1\ \{\mathsf{x}=10\}$$

$$\{\mathbf{true}\}\ \mathbf{while}\ \mathsf{x}\neq 10\ \mathbf{do}\ \mathsf{x}:=\mathsf{x}+1\ \{\mathsf{x}=10\}$$

$$\{\mathsf{x}>10\}\ \mathbf{while}\ \mathsf{x}\neq 10\ \mathbf{do}\ \mathsf{x}:=\mathsf{x}+1\ \{\mathbf{false}\}$$

$$\{\mathbf{emp}\}\ \mathsf{x}:=\mathbf{cons}(1,2)\ \{\mathsf{x}\mapsto 1,2\}$$

$$\{\mathsf{x}\mapsto 1,2\}\ \mathsf{y}:=[\mathsf{x}]\ ;[\mathsf{x}+1]:=3\ \{\mathsf{x}\mapsto 1,3\wedge \mathsf{y}=1\}$$

Examples of Total Correctness Specifications

$$[\,\mathsf{x}-\mathsf{y}>3\,]\ \mathsf{x}:=\mathsf{x}-\mathsf{y}\ [\,\mathsf{x}>3\,]$$

$$[\,\mathsf{x}+\mathsf{y}\geq 17\,]\ \mathsf{x}:=\mathsf{x}+10\ [\,\mathsf{x}+\mathsf{y}\geq 27\,]$$

$$[\,\mathsf{x}\leq 10\,]\ \mathbf{while}\ \mathsf{x}\neq 10\ \mathbf{do}\ \mathsf{x}:=\mathsf{x}+1\ [\,\mathsf{x}=10\,]$$

$$[\,\mathbf{emp}\,]\ \mathsf{x}:=\mathbf{cons}(1,2)\ [\,\mathsf{x}\mapsto 1,2\,]$$

$$[\,\mathsf{x}\mapsto 1,2\,]\ \mathsf{y}:=[\mathsf{x}]\ ;[\mathsf{x}+1]:=3\ [\,\mathsf{x}\mapsto 1,3\wedge \mathsf{y}=1\,]$$

Inference Rules for Specifications:

Assignment (AS)

$$\frac{}{\{p/v \to e\}\; v := e\; \{p\}} \qquad \frac{}{[\, p/v \to e\,]\; v := e\; [\, p\,]}$$

Instances

$$\frac{}{\{\mathsf{x} - \mathsf{y} > 3\}\; \mathsf{x} := \mathsf{x} - \mathsf{y}\; \{\mathsf{x} > 3\}}$$

$$\frac{}{\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\; \mathsf{x} := 2 \times \mathsf{y}\; \{\mathsf{x} - \mathsf{y} > 3\}}$$

Inference Rules for Specifications:

Sequential Composition (SQ)

$$\frac{\{p\}\ c_1\ \{q\} \qquad \{q\}\ c_2\ \{r\}}{\{p\}\ c_1\, ;\, c_2\ \{r\}} \qquad \frac{[\,p\,]\ c_1\ [\,q\,] \qquad [\,q\,]\ c_2\ [\,r\,]}{[\,p\,]\ c_1\, ;\, c_2\ [\,r\,]}$$

An Instance

$$\frac{\begin{array}{r}\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\ \{\mathsf{x} - \mathsf{y} > 3\} \\ \{\mathsf{x} - \mathsf{y} > 3\}\ \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}\end{array}}{\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\, ;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}}$$

Note

Except for **while** commands and recursive procedures, reasoning for partial and total correctness will be the same, and we will henceforth only give rules for partial correctness.

Inference Rules for Specifications:

Strengthening Precedent (SP)

$$\frac{p \Rightarrow q \qquad \{q\}\ c\ \{r\}}{\{p\}\ c\ \{r\}}$$

An Instance

$$\frac{\begin{array}{l}\mathsf{y} > 3 \Rightarrow (2 \times \mathsf{y}) - \mathsf{y} > 3 \\ \qquad \{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}\end{array}}{\{\mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}}$$

Weakening Consequent (WC)

$$\frac{\{p\}\ c\ \{q\} \qquad q \Rightarrow r}{\{p\}\ c\ \{r\}}$$

An Instance

$$\frac{\{\mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\} \qquad \mathsf{x} > 3 \Rightarrow \mathsf{x} \geq 4}{\{\mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} \geq 4\}}$$

The implications in (SP) and (WC) are called *verification conditions* (VC).

A Simple Proof

1. $\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\ \{\mathsf{x} - \mathsf{y} > 3\}$ (AS)
2. $\{\mathsf{x} - \mathsf{y} > 3\}\ \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}$ (AS)
3. $\{(2 \times \mathsf{y}) - \mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}$ (SQ,1,2)
4. $\mathsf{y} > 3 \Rightarrow (2 \times \mathsf{y}) - \mathsf{y} > 3$ (VC)
5. $\{\mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} > 3\}$ (SP,4,3)
6. $\mathsf{x} > 3 \Rightarrow \mathsf{x} \geq 4$ (VC)
7. $\{\mathsf{y} > 3\}\ \mathsf{x} := 2 \times \mathsf{y}\,;\, \mathsf{x} := \mathsf{x} - \mathsf{y}\ \{\mathsf{x} \geq 4\}$ (WC,5,6)

A More Realistic Proof

1. $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (AS)
2. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{f} := \mathsf{f}+\mathsf{t}$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$ (AS)
3. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{f} := \mathsf{f}+\mathsf{t}\,;\mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (SQ,1,2)
4. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{g} := \mathsf{f}$
   $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{g} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$ (AS)
5. $\{\mathsf{f}+\mathsf{t} = \mathrm{fib}(\mathsf{k}+1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k}+1)-1) \wedge \mathsf{k}+1 \leq \mathsf{n}\}$
   $\mathsf{g} := \mathsf{f}\,;\mathsf{f} := \mathsf{f}+\mathsf{t}\,;\mathsf{k} := \mathsf{k}+1$
   $\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}$ (SQ,3,4)

A More Realistic Proof (continued)

6. $\{\mathsf{f}+\mathsf{g}=\mathrm{fib}(\mathsf{k}+1)\wedge\mathsf{f}=\mathrm{fib}((\mathsf{k}+1)-1)\wedge\mathsf{k}+1\le\mathsf{n}\}$
   $\mathsf{t}:=\mathsf{g}$
   $\{\mathsf{f}+\mathsf{t}=\mathrm{fib}(\mathsf{k}+1)\wedge\mathsf{f}=\mathrm{fib}((\mathsf{k}+1)-1)\wedge\mathsf{k}+1\le\mathsf{n}\}$ (AS)

7. $\{\mathsf{f}+\mathsf{g}=\mathrm{fib}(\mathsf{k}+1)\wedge\mathsf{f}=\mathrm{fib}((\mathsf{k}+1)-1)\wedge\mathsf{k}+1\le\mathsf{n}\}$
   $\mathsf{t}:=\mathsf{g}\,;\,\mathsf{g}:=\mathsf{f}\,;\,\mathsf{f}:=\mathsf{f}+\mathsf{t}\,;\,\mathsf{k}:=\mathsf{k}+1$
   $\{\mathsf{f}=\mathrm{fib}(\mathsf{k})\wedge\mathsf{g}=\mathrm{fib}(\mathsf{k}-1)\wedge\mathsf{k}\le\mathsf{n}\}$ (SQ,5,6)

8. $(\mathsf{f}=\mathrm{fib}(\mathsf{k})\wedge\mathsf{g}=\mathrm{fib}(\mathsf{k}-1)\wedge\mathsf{k}\le\mathsf{n}\wedge\mathsf{k}\neq\mathsf{n})\Rightarrow$
   $(\mathsf{f}+\mathsf{g}=\mathrm{fib}(\mathsf{k}+1)\wedge\mathsf{f}=\mathrm{fib}((\mathsf{k}+1)-1)\wedge\mathsf{k}+1\le\mathsf{n})$
   (VC*)

9. $\{\mathsf{f}=\mathrm{fib}(\mathsf{k})\wedge\mathsf{g}=\mathrm{fib}(\mathsf{k}-1)\wedge\mathsf{k}\le\mathsf{n}\wedge\mathsf{k}\neq\mathsf{n}\}$
   $\mathsf{t}:=\mathsf{g}\,;\,\mathsf{g}:=\mathsf{f}\,;\,\mathsf{f}:=\mathsf{f}+\mathsf{t}\,;\,\mathsf{k}:=\mathsf{k}+1$
   $\{\mathsf{f}=\mathrm{fib}(\mathsf{k})\wedge\mathsf{g}=\mathrm{fib}(\mathsf{k}-1)\wedge\mathsf{k}\le\mathsf{n}\}$ (SP,7,8)

*since

$$\mathrm{fib}(0)=0 \qquad \mathrm{fib}(1)=1 \qquad \mathrm{fib}(i)=\mathrm{fib}(i-1)+\mathrm{fib}(i-2).$$

Inference Rules for Specifications:

An Alternative "Forward" Rule for Assignment (ASFW)

$$\overline{\{p\}\ v := e\ \{\exists v'.\ v = e' \wedge p'\}}$$

where $v' \notin \{v\} \cup \mathrm{FV}(e) \cup \mathrm{FV}(p)$, e' is $e/v \to v'$, and p' is $p/v \to v'$. The quantifier can be omitted when v does not occur in e or p.

The problem with this rule is the accumulation of quantifiers. For example, when we use it to prove

$$\begin{aligned}&\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\}\\ &\mathsf{t} := \mathsf{g}\ ;\ \mathsf{g} := \mathsf{f}\ ;\ \mathsf{f} := \mathsf{f} + \mathsf{t}\ ;\ \mathsf{k} := \mathsf{k} + 1\\ &\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\},\end{aligned}$$

we get the verification condition

$$\begin{aligned}&(\exists \mathsf{k}'.\ \exists \mathsf{f}'.\ \exists \mathsf{g}'.\ \mathsf{k} = \mathsf{k}' + 1 \wedge \mathsf{f} = \mathsf{f}' + \mathsf{t} \wedge \mathsf{g} = \mathsf{f}' \wedge \mathsf{t} = \mathsf{g}'\\ &\qquad\qquad \wedge\, \mathsf{f}' = \mathrm{fib}(\mathsf{k}') \wedge \mathsf{g}' = \mathrm{fib}(\mathsf{k}'-1) \wedge \mathsf{k}' \leq \mathsf{n} \wedge \mathsf{k}' \neq \mathsf{n}) \Rightarrow\\ &\qquad \mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}.\end{aligned}$$

Inference Rules for Specifications:

Partial Correctness of **while** (WHP)

$$\frac{\{i \wedge b\}\ c\ \{i\}}{\{i\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{i \wedge \neg\, b\}}$$

Here i is the *invariant*.

An Instance

$$\frac{\begin{array}{l}\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}\} \\ \mathsf{t} := \mathsf{g}\ ;\ \mathsf{g} := \mathsf{f}\ ;\ \mathsf{f} := \mathsf{f} + \mathsf{t}\ ;\ \mathsf{k} := \mathsf{k} + 1 \\ \{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\}\end{array}}{\begin{array}{l}\{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}\} \\ \mathbf{while}\ \mathsf{k} \neq \mathsf{n}\ \mathbf{do}\ (\mathsf{t} := \mathsf{g}\ ;\ \mathsf{g} := \mathsf{f}\ ;\ \mathsf{f} := \mathsf{f} + \mathsf{t}\ ;\ \mathsf{k} := \mathsf{k} + 1) \\ \{\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\, \mathsf{k} \neq \mathsf{n}\}\end{array}}$$

Inference Rules for Specifications:

Total Correctness of **while** (WHT)

$$\frac{[\, i \wedge b \wedge e = v_0 \,]\; c\; [\, i \wedge e < v_0 \,] \qquad (i \wedge b) \Rightarrow e \geq 0}{[\, i \,]\; \textbf{while}\; b\; \textbf{do}\; c\; [\, i \wedge \neg\, b \,]}$$

when v_0 does not occur free in i, b, c, or e.
Here i is the *invariant*, e is the *variant*, and v_0 is a *ghost variable*.

An Instance

$$\frac{\begin{array}{l} [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n} \wedge \mathsf{n} - \mathsf{k} = \mathsf{v}_0] \\ \mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k} + 1 \\ [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{n} - \mathsf{k} < \mathsf{v}_0] \\ \\ (\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n}) \Rightarrow \mathsf{n} - \mathsf{k} \geq 0 \end{array}}{\begin{array}{l} [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n}] \\ \textbf{while}\; \mathsf{k} \neq \mathsf{n}\; \textbf{do}\; (\mathsf{t} := \mathsf{g}\,;\, \mathsf{g} := \mathsf{f}\,;\, \mathsf{f} := \mathsf{f} + \mathsf{t}\,;\, \mathsf{k} := \mathsf{k} + 1) \\ [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k}-1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\, \mathsf{k} \neq \mathsf{n}] \end{array}}$$

Inference Rules for Specifications:

skip (SK)

$$\frac{}{\{p\}\ \mathbf{skip}\ \{p\}}$$

Conditional (CD)

$$\frac{\{p \wedge b\}\ c_1\ \{q\} \qquad \{p \wedge \neg\ b\}\ c_2\ \{q\}}{\{p\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}}$$

An Alternative Rule for Conditionals (CDALT)

$$\frac{\{p_1\}\ c_1\ \{q\} \qquad \{p_2\}\ c_2\ \{q\}}{\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}}$$

Inference Rules for Specifications:

Variable Declaration (DC)

$$\frac{\{p\}\ c\ \{q\}}{\{p\}\ \mathbf{newvar}\ v\ \mathbf{in}\ c\ \{q\}}$$

when v does not occur free in p or q.

Here the requirement on the declared variable v formalizes the concept of *locality*, i.e., that the value of v when c begins execution has no effect on this execution, and that the value of v when c finishes execution has no effect on the rest of the program.

Locality is a property of the specification of a command rather than just the command itself, since it depends on the use to which the command may be put. For example, the same command satisfies both of the specifications

$$\{\mathbf{true}\}\ \mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{z} := \mathsf{t} \times \mathsf{t}\ \{\mathsf{z} = (\mathsf{x} + \mathsf{y})^2\}$$

$$\{\mathbf{true}\}\ \mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{z} := \mathsf{t} \times \mathsf{t}\ \{\mathsf{z} = (\mathsf{x} + \mathsf{y})^2 \wedge \mathsf{t} = \mathsf{x} + \mathsf{y}\}.$$

But the variable t is local only in the first case, so that **newvar** t **in** $(\mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{z} := \mathsf{t} \times \mathsf{t})$ meets only the first specification:

$$\{\mathbf{true}\}\ \mathbf{newvar}\ \mathsf{t}\ \mathbf{in}\ (\mathsf{t} := \mathsf{x} + \mathsf{y}\ ;\ \mathsf{z} := \mathsf{t} \times \mathsf{t})\ \{\mathsf{z} = (\mathsf{x} + \mathsf{y})^2\}.$$

Annotated Specifications

In an *annotated specification*, additional assertions called *annotations* are placed within the command in such a way that the proof of the specification can be reconstructed from the annotations. To determine this reconstruction, we must use rules in which the premises are completely determined by the conclusion. This requires us to change the following rules:

Sequential Composition (SQAN)

$$\frac{\{p\}\ c_1\ \{q\} \qquad \{q\}\ c_2\ \{r\}}{\{p\}\ c_1\ ;\{q\}\ c_2\ \{r\}}$$

Strengthening Precedent (SPAN)

$$\frac{p \Rightarrow q \qquad \{q\}\ c\ \{r\}}{\{p\}\ \{q\}\ c\ \{r\}}$$

Weakening Consequent (WCAN)

$$\frac{\{p\}\ c\ \{q\} \qquad q \Rightarrow r}{\{p\}\ c\ \{q\}\ \{r\}}$$

Total Correctness of **while**

$$\frac{[\, i \wedge b \wedge e = v_0 \,]\ c\ [\, i \wedge e < v_0 \,] \qquad (i \wedge b) \Rightarrow e \geq 0}{[\, i \,]\ \mathbf{while}\ [\mathbf{vrnt}\colon e]\ b\ \mathbf{do}\ c\ [\, i \wedge \neg\ b \,]}$$

when v_0 does not occur free in i, b, c, or e.

Annotated Specifications (continued)

To reduce the number of unnecessary annotations, it is useful to use the following derived inference rules:

Assignment (ASAN)

$$\frac{p_0 \Rightarrow (p/v \to e)}{\{p_0\}\ v := e\ \{p\}} \qquad \frac{\{p_0\}\ c\ \{p/v \to e\}}{\{p_0\}\ c\ ;\ v := e\ \{p\}}$$

skip (SKAN)

$$\frac{p_0 \Rightarrow p}{\{p_0\}\ \mathbf{skip}\ \{p\}}$$

Partial Correctness of **while** (WHPAN)

$$\frac{\{i \wedge b\}\ c\ \{i\} \qquad (i \wedge \neg\, b) \Rightarrow p}{\{i\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{p\}}$$

Total Correctness of **while** (WHTAN)

$$\frac{[\, i \wedge b \wedge e = v_0\,]\ c\ [\, i \wedge e < v_0\,] \quad (i \wedge b) \Rightarrow e \geq 0 \quad (i \wedge \neg\ b) \Rightarrow p}{[\, i\,]\ \mathbf{while}\ [\mathbf{vrnt}\colon e]\ b\ \mathbf{do}\ c\ [\, p\,]}$$

when v_0 does not occur free in i, b, c, or e.

An Annotated Specification

$[\mathsf{n} \geq 0]$

$\textbf{if } \mathsf{n} = 0 \textbf{ then } \mathsf{f} := 0 \textbf{ else}$

$\quad \textbf{newvar } \mathsf{g} \textbf{ in newvar } \mathsf{k} \textbf{ in}$

$\qquad (\mathsf{k} := 1 \,;\, \mathsf{g} := 0 \,;\, \mathsf{f} := 1 \,;$

$\qquad [\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k} - 1) \wedge \mathsf{k} \leq \mathsf{n}]$

$\qquad \textbf{while } [\textbf{vrnt}\!: \mathsf{n} - \mathsf{k}]\, \mathsf{k} \neq \mathsf{n} \textbf{ do}$

$\qquad\quad \textbf{newvar } \mathsf{t} \textbf{ in } (\mathsf{t} := \mathsf{g} \,;\, \mathsf{g} := \mathsf{f} \,;\, \mathsf{f} := \mathsf{f} + \mathsf{t} \,;\, \mathsf{k} := \mathsf{k} + 1))$

$[\mathsf{f} = \mathrm{fib}(\mathsf{n})]$

Verification Conditions

1. $\mathsf{n} \geq 0 \wedge \mathsf{n} = 0 \Rightarrow 0 = \mathrm{fib}(\mathsf{n})$
2. $\mathsf{n} \geq 0 \wedge \neg\, \mathsf{n} = 0 \Rightarrow 1 = \mathrm{fib}(1) \wedge 0 = \mathrm{fib}(1 - 1) \wedge 1 \leq \mathsf{n}$
3. $\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k} - 1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n} \wedge \mathsf{n} - \mathsf{k} = \mathsf{v}_0 \Rightarrow$
   $\quad \mathsf{f} + \mathsf{g} = \mathrm{fib}(\mathsf{k} + 1) \wedge \mathsf{f} = \mathrm{fib}((\mathsf{k} + 1) - 1) \wedge \mathsf{k} + 1 \leq \mathsf{n}$
   $\qquad \wedge\, \mathsf{n} - (\mathsf{k} + 1) < \mathsf{v}_0$
4. $\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k} - 1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \mathsf{k} \neq \mathsf{n} \Rightarrow \mathsf{n} - \mathsf{k} \geq 0$
5. $\mathsf{f} = \mathrm{fib}(\mathsf{k}) \wedge \mathsf{g} = \mathrm{fib}(\mathsf{k} - 1) \wedge \mathsf{k} \leq \mathsf{n} \wedge \neg\, \mathsf{k} \neq \mathsf{n} \Rightarrow \mathsf{f} = \mathrm{fib}(\mathsf{n})$

Interderivability of the Assignment Rules

$$\frac{}{\{p/v \rightarrow e\}\; v := e\; \{p\}}$$

$$\frac{}{\{p\}\; v := e\; \{\exists v'.\; v = e' \wedge p'\}}$$

where $v' \notin \{v\} \cup \mathrm{FV}(e) \cup \mathrm{FV}(p)$, e' is $e/v \rightarrow v'$, and p' is $p/v \rightarrow v'$.

$$\begin{array}{l}
\{p\} \\
\{e = e \wedge p\} \\
\{(e = e' \wedge p')/v' \rightarrow v\} \\
\{\exists v'.\; e = e' \wedge p'\} \\
v := e \\
\{\exists v'.\; v = e' \wedge p'\}
\end{array}$$

$$\begin{array}{l}
\{p/v \rightarrow e\} \\
v := e \\
\{\exists v'.\; v = e' \wedge ((p/v \rightarrow e)/v \rightarrow v')\} \\
\{\exists v'.\; v = e' \wedge (p/v \rightarrow e')\} \\
\{\exists v'.\; v = e' \wedge p\} \\
\{\exists v'.\; p\} \\
\{p\}
\end{array}$$

Interderivability of the Conditional Rules

$$\frac{\{p \wedge b\}\ c_1\ \{q\} \qquad \{p \wedge \neg\ b\}\ c_2\ \{q\}}{\{p\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}}$$

$$\frac{\{p_1\}\ c_1\ \{q\} \qquad \{p_2\}\ c_2\ \{q\}}{\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}}$$

Assume the premises of the second rule. Then

$\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2) \wedge b\}$
$\{p_1\}$
c_1
$\{q\}$

and

$\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2) \wedge \neg\ b\}$
$\{p_2\}$
c_2
$\{q\}$,

and the first rule gives the conclusion of the second rule:

$\{(b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2)\}$
$\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2$
$\{q\}$.

Interderivability of the Conditional Rules (continued)

$$\frac{\{p \wedge b\}\ c_1\ \{q\} \qquad \{p \wedge \neg\, b\}\ c_2\ \{q\}}{\{p\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}}$$

$$\frac{\{p_1\}\ c_1\ \{q\} \qquad \{p_2\}\ c_2\ \{q\}}{\{(b \Rightarrow p_1) \wedge (\neg\, b \Rightarrow p_2)\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{q\}}$$

Assume the premises of the first rule. Then the second rule gives the main step in

$$\begin{array}{l}
\{p\} \\
\{(b \Rightarrow p \wedge b) \wedge (\neg\, b \Rightarrow p \wedge \neg\, b)\} \\
\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 \\
\{q\},
\end{array}$$

which is the conclusion of the second rule.

Weakest (Liberal) Preconditions

$[\![\mathbf{wlp}(c,q)]\!]_{\text{assert}}(s,h)$ iff

$\neg\,(c,(s,h) \rightsquigarrow^* \mathbf{abort})$

and $(\forall (s',h') \in \text{States}.\ c,(s,h) \rightsquigarrow^* (s',h')$ implies $[\![q]\!]_{\text{assert}} s'\, h')$.

$[\![\mathbf{wp}(c,q)]\!]_{\text{assert}}(s,h)$ iff

$\neg\,(c,(s,h) \rightsquigarrow^* \mathbf{abort})$

and $\neg\,(c,(s,h)\uparrow)$

and $(\forall (s',h') \in \text{States}.\ c,(s,h) \rightsquigarrow^* (s',h')$ implies $[\![q]\!]_{\text{assert}} s'\, h')$.

Then

$$\{p\}\ c\ \{q\} \text{ holds iff } p \Rightarrow \mathbf{wlp}(c,q) \text{ is valid.}$$

$$[\,p\,]\ c\ [\,q\,] \text{ holds iff } p \Rightarrow \mathbf{wp}(c,q) \text{ is valid.}$$

Also, we define

$$\{p\}_{\text{wlp}}\ c\ \{q\} \text{ holds iff } p \Leftrightarrow \mathbf{wlp}(c,q) \text{ is valid.}$$

$$[\,p\,]_{\text{wp}}\ c\ [\,q\,] \text{ holds iff } p \Leftrightarrow \mathbf{wp}(c,q) \text{ is valid.}$$

Some Rules for Weakest Preconditions:

Assignment (ASWP)

$$\mathbf{wp}(v := e, p) \Leftrightarrow (p/v \to e)$$

$$\frac{}{[\, p/v \to e \,]_{\mathrm{wp}}\; v := e\; [\, p \,]}$$

skip (SKWP)

$$\mathbf{wp}(\mathbf{skip}, p) \Leftrightarrow p$$

$$\frac{}{[\, p \,]_{\mathrm{wp}}\; \mathbf{skip}\; [\, p \,]}$$

Sequential Composition (SQWP)

$$\mathbf{wp}(c_1\,;\, c_2, r) \Leftrightarrow \mathbf{wp}\Big(c_1, \mathbf{wp}(c_2, r)\Big)$$

$$\frac{[\, p \,]_{\mathrm{wp}}\; c_1\; [\, q \,] \qquad [\, q \,]_{\mathrm{wp}}\; c_2\; [\, r \,]}{[\, p \,]_{\mathrm{wp}}\; c_1\,;\, c_2\; [\, r \,]}$$

Conditional (CDWP)

$$\mathbf{wp}(\mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2, q) \Leftrightarrow \Big(b \Rightarrow \mathbf{wp}(c_1, q)\Big) \wedge \Big(\neg\ b \Rightarrow \mathbf{wp}(c_2, q)\Big)$$

$$\frac{[\, p_1 \,]_{\mathrm{wp}}\; c_1\; [\, q \,] \qquad [\, p_2 \,]_{\mathrm{wp}}\; c_2\; [\, q \,]}{[\, (b \Rightarrow p_1) \wedge (\neg\ b \Rightarrow p_2) \,]_{\mathrm{wp}}\; \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\; [\, q \,]}$$

Structural Inference Rules:

Renaming (RN)

$$\frac{\{p\}\ c\ \{q\}}{\{p'\}\ c'\ \{q'\},}$$

where p', c', and q' are obtained from p, c, and q by zero or more renamings of bound variables.

Free Variable Substitution (FVS)

$$\frac{\{p\}\ c\ \{q\}}{(\{p\}\ c\ \{q\})/v_1 \to e_1,\ \ldots\ ,\ v_n \to e_n,}$$

where $v_1, \ldots, v_n$ are the variables occurring free in p, c, or q, and, if v_i is assigned by c, then e_i is a variable that does not occur free in any other e_j.

For example, in

$$\{\mathsf{x} = \mathsf{y}\}\ \mathsf{x} := \mathsf{x} + \mathsf{y}\ \{\mathsf{x} = 2 \times \mathsf{y}\},$$

one can substitute $\mathsf{x}, \mathsf{y}\ \to\ \mathsf{z}, 2 \times \mathsf{w} - 1$ to infer

$$\{\mathsf{z} = 2 \times \mathsf{w} - 1\}\ \mathsf{z} := \mathsf{z} + 2 \times \mathsf{w} - 1\ \{\mathsf{z} = 2 \times (2 \times \mathsf{w} - 1)\}.$$

But one cannot substitute $\mathsf{x}, \mathsf{y}\ \to\ \mathsf{z}, 2 \times \mathsf{z} - 1$ to infer

$$\{\mathsf{z} = 2 \times \mathsf{z} - 1\}\ \mathsf{z} := \mathsf{z} + 2 \times \mathsf{z} - 1\ \{\mathsf{z} = 2 \times (2 \times \mathsf{z} - 1)\}.$$

Free Variable Substitution (continued)

$$\frac{\{p\}\ c\ \{q\}}{(\{p\}\ c\ \{q\})/v_1 \to e_1,\ \ldots\ ,\ v_n \to e_n,}$$

where $v_1, \ldots, v_n$ are the variables occurring free in p, c, or q, and, if v_i is assigned by c, then e_i is a variable that does not occur free in any other e_j.

Proposition 7 *The free variable substitution rule is sound.*

PROOF Let δ abbreviate the substitution

$$v_1 \to e_1, \ldots, v_n \to e_n,$$

and for any store s such that $\mathrm{FV}(e_1) \cup \cdots \cup \mathrm{FV}(e_n) \subseteq \mathrm{dom}\, s$, let

$$\widehat{s} = [\, v_1\colon [\![e_1]\!]_{\mathrm{exp}} s \mid \ldots \mid v_n\colon [\![e_n]\!]_{\mathrm{exp}} s \,].$$

- Assume $\{p\}\ c\ \{q\}$ and $[\![p/\delta]\!]_{\mathrm{assert}} s\, h$. By Proposition 6, $[\![p]\!]_{\mathrm{assert}} \widehat{s}\, h$.

- Suppose $\langle c/\delta, (s, h)\rangle \rightsquigarrow^* \textbf{abort}$. By Proposition 4, we would have $\langle c, (\widehat{s}, h)\rangle \rightsquigarrow^* \textbf{abort}$, which contradicts $\{p\}\ c\ \{q\}$ and $[\![p]\!]_{\mathrm{assert}} \widehat{s}\, h$.

- Suppose $\langle c/\delta, (s, h)\rangle \rightsquigarrow^* (s', h')$. By Proposition 4, we have $\langle c, (\widehat{s}, h)\rangle \rightsquigarrow^* (\widehat{s'}, h')$, so that $\{p\}\ c\ \{q\}$ and $[\![p]\!]_{\mathrm{assert}} \widehat{s}\, h$ give $[\![q]\!]_{\mathrm{assert}} \widehat{s'} h'$. Then, by Proposition 6, $[\![q/\delta]\!]_{\mathrm{assert}} s' h'$.

END OF PROOF

Structural Inference Rules:

Disjunction of Assertions (DA)

$$\frac{\{p_1\}\ c\ \{q_1\} \qquad \{p_2\}\ c\ \{q_2\}}{\{p_1 \vee p_2\}\ c\ \{q_1 \vee q_2\}}$$

Conjunction of Assertions (CA)

$$\frac{\{p_1\}\ c\ \{q_1\} \qquad \{p_2\}\ c\ \{q_2\}}{\{p_1 \wedge p_2\}\ c\ \{q_1 \wedge q_2\}}$$

Existential Quantification of Assertions (EQ)

$$\frac{\{p\}\ c\ \{q\}}{\{\exists v.\ p\}\ c\ \{\exists v.\ q\}},$$

where v is not free in c.

Universal Quantification of Assertions (UQ)

$$\frac{\{p\}\ c\ \{q\}}{\{\forall v.\ p\}\ c\ \{\forall v.\ q\}},$$

where v is not free in c.

Structural Inference Rules

The following rule is sound for Hoare logic, but not for our extension:

The Unsound Rule of Constancy

$$\frac{\{p\}\ c\ \{q\}}{\{p \wedge r\}\ c\ \{q \wedge r\}}$$

when no variable occurs both free in r and assigned by c.

Counterexamples

$$\frac{\{\mathsf{x} = \mathsf{y} \wedge \mathsf{x} \mapsto 3\}\ [\mathsf{x}] := 4\ \{\mathsf{x} = \mathsf{y} \wedge \mathsf{x} \mapsto 4\}}{\{\mathsf{x} = \mathsf{y} \wedge \mathsf{x} \mapsto 3 \wedge \mathsf{y} \mapsto 3\}\ [\mathsf{x}] := 4\ \{\mathsf{x} = \mathsf{y} \wedge \mathsf{x} \mapsto 4 \wedge \mathsf{y} \mapsto 3\}}$$

$$\frac{\{\mathsf{x} \mapsto 3\}\ [\mathsf{x}] := 4\ \{\mathsf{x} \mapsto 4\}}{\{\mathsf{x} \mapsto 3 \wedge \mathsf{x} \mapsto 3\}\ [\mathsf{x}] := 4\ \{\mathsf{x} \mapsto 4 \wedge \mathsf{x} \mapsto 3\}}$$

Structural Inference Rules:

The Frame Rule (FR)

$$\frac{\{p\}\ c\ \{q\}}{\{p \ast r\}\ c\ \{q \ast r\}}$$

when no variable occurs both free in r and assigned by c.

Proposition 8 *The frame rule is sound.*

PROOF

- Assume $\{p\}\ c\ \{q\}$, and $[\![p \ast r]\!]_{\text{assert}} s\, h$. Then there are h_0, h_1 such that $h_0 \perp h_1$, $h = h_0 \cdot h_1$, $[\![p]\!]_{\text{assert}} s\, h_0$ and $[\![r]\!]_{\text{assert}} s\, h_1$.
- Suppose $\langle c, (s, h)\rangle \leadsto^* \textbf{abort}$. Then, by Proposition 5, we would have $\langle c, (s, h_0)\rangle \leadsto^* \textbf{abort}$, which contradicts $\{p\}\ c\ \{q\}$ and $[\![p]\!]_{\text{assert}} s\, h_0$.
- Suppose $\langle c, (s, h)\rangle \leadsto^* (s', h')$. Again, $\langle c, (s, h_0)\rangle \leadsto^* \textbf{abort}$ would contradict $\{p\}\ c\ \{q\}$ and $[\![p]\!]_{\text{assert}} s\, h_0$, so that, by Proposition 5, $\langle c, (s, h_0)\rangle \leadsto^* (s', h'_0)$, where $h'_0 \perp h_1$ and $h' = h'_0 \cdot h_1$. Then $\{p\}\ c\ \{q\}$ and $[\![p]\!]_{\text{assert}} s\, h_0$ imply that $[\![q]\!]_{\text{assert}} s' h'_0$.

  Since $\langle c, (s, h)\rangle \leadsto^* (s', h')$, Proposition 3 gives $s\, v = s' v$ for all v that are not assigned by c. Then, since these include the free variables of r, $[\![r]\!]_{\text{assert}} s\, h_1$ implies that $[\![r]\!]_{\text{assert}} s' h_1$. Thus $[\![q \ast r]\!]_{\text{assert}} s' h$.

END OF PROOF

Inference Rules for Mutation

The small form (MUSM):

$$\frac{}{\{e \mapsto -\}\ [e] := e'\ \{e \mapsto e'\}.}$$

The large form (MULG):

$$\frac{}{\{(e \mapsto -) \ast r\}\ [e] := e'\ \{(e \mapsto e') \ast r\}.}$$

The backward-reasoning form (MUBR):

$$\frac{}{\{(e \mapsto -) \ast ((e \mapsto e') \mathrel{-\!\!\ast} p)\}\ [e] := e'\ \{p\}.}$$

One can derive (MULG) from (MUSM) by using the frame rule; one can go in the opposite direction by taking r to be **emp**.

One can derive (MUBR) from (MULG) by taking r to be $(e \mapsto e') \mathrel{-\!\!\ast} p$ and using the law $q \ast (q \mathrel{-\!\!\ast} p) \Rightarrow p$:

$$\begin{array}{l}
\{(e \mapsto -) \ast ((e \mapsto e') \mathrel{-\!\!\ast} p)\} \\
[e] := e' \\
\{(e \mapsto e') \ast ((e \mapsto e') \mathrel{-\!\!\ast} p)\} \\
\{p\}.
\end{array}$$

One can go in the opposite direction by taking p to be $(e \mapsto e') \ast r$ and using the law $(p \ast r) \Rightarrow (p \ast (q \mathrel{-\!\!\ast} (q \ast r)))$:

$$\begin{array}{l}
\{(e \mapsto -) \ast r\} \\
\{(e \mapsto -) \ast ((e \mapsto e') \mathrel{-\!\!\ast} ((e \mapsto e') \ast r))\} \\
[e] := e' \\
\{(e \mapsto e') \ast r\}.
\end{array}$$

Inference Rules for Noninterfering Allocation

The small form (CONSNISM):

$$\frac{}{\{\mathbf{emp}\}\ v := \mathbf{cons}(e_0, \ldots, e_{n-1})\ \{v \mapsto e_0, \ldots, e_{n-1}\},}$$

where $v \notin \mathrm{FV}(e_0, \ldots, e_{n-1})$.

The large form (CONSNILG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(e_0, \ldots, e_{n-1})\ \{(v \mapsto e_0, \ldots, e_{n-1})\ *\ r\},}$$

where $v \notin \mathrm{FV}(e_0, \ldots, e_{n-1}, r)$.

One can derive (CONSNILG) from (CONSNISM) by using the frame rule (FR); one can go in the opposite direction by taking r to be **emp**.

Inference Rules for General Allocation

The small form (CONSSM):

$$\frac{}{\{v = v' \wedge \mathbf{emp}\}\ v := \mathbf{cons}(e_0, \ldots, e_{n-1})\ \{v \mapsto e'_0, \ldots, e'_{n-1}\}.}$$

where v' is distinct from v, and e'_i denotes $e_i/v \to v'$.

The large form (CONSLG):

$$\frac{}{\{r\}\ v := \mathbf{cons}(e_0, \ldots, e_{n-1})\ \{\exists v'.\ (v \mapsto e'_0, \ldots, e'_{n-1})\ *\ r'\},}$$

where v' is distinct from v, $v' \notin \mathrm{FV}(e_0, \ldots, e_{n-1}, r)$, e'_i denotes $e_i/v \to v'$, and r' denotes $r/v \to v'$.

The backward-reasoning form (CONSBR):

$$\frac{}{\{\forall v''.\ (v'' \mapsto e_0, \ldots, e_{n-1}) \mathrel{-\!\!*} p''\}\ v := \mathbf{cons}(e_0, \ldots, e_{n-1})\ \{p\},}$$

where v'' is distinct from v, $v'' \notin \mathrm{FV}(e_0, \ldots, e_{n-1}, p)$, and p'' denotes $p/v \to v''$.

Inference Rules for General Allocation (continued)

Let $\overline{e}$ abbreviate $e_0, \ldots, e_{n-1}$, $\overline{e}'$ abbreviate $e'_0, \ldots, e'_{n-1}$, and similarly for e''. One can derive (CONSLG) from (CONSSM) by using (FR) and (EQ):

$$
\begin{array}{ll}
\{v = v' \wedge r'\} & \{r\} \\
\{v = v' \wedge (\mathbf{emp} \ * \ r')\} & \{\exists v'.\ v = v' \wedge r\} \\
\{(v = v' \wedge \mathbf{emp}) \ * \ r'\} & \{\exists v'.\ v = v' \wedge r'\} \\
v := \mathbf{cons}(\overline{e}) & v := \mathbf{cons}(\overline{e}) \\
\{(v \mapsto \overline{e}') \ * \ r'\}, & \{\exists v'.\ (v \mapsto \overline{e}') \ * \ r'\}.
\end{array}
$$

To go in the other direction:

$$
\begin{array}{l}
\{v = v' \wedge \mathbf{emp}\} \\
v := \mathbf{cons}(\overline{e}) \\
\{\exists v''.\ (v \mapsto \overline{e}'') \ * \ (v'' = v' \wedge \mathbf{emp})\} \\
\{\exists v''.\ ((v \mapsto \overline{e}'') \ * \ \mathbf{emp}) \wedge v'' = v'\} \\
\{\exists v''.\ (v \mapsto \overline{e}'') \wedge v'' = v'\} \\
\{\exists v''.\ (v \mapsto \overline{e}') \wedge v'' = v'\} \\
\{v \mapsto \overline{e}'\}.
\end{array}
$$

Inference Rules for General Allocation (continued)

In order to derive (CONSBR) from (CONSLG), we choose $v' \notin \mathrm{FV}(e_0, \ldots, e_{n-1}, p)$ to be distinct from v and v'', take r to be $\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} p''$, and use various laws about quantifiers, as well as $q \ast (q \mathrel{-\!\!*} p) \Rightarrow p$:

$$
\begin{array}{l}
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} p''\} \\
v := \mathbf{cons}(\overline{e}) \\
\{\exists v'.\ (v \mapsto \overline{e}') \ast (\forall v''.\ (v'' \mapsto \overline{e}') \mathrel{-\!\!*} p'')\} \\
\{\exists v'.\ (v \mapsto \overline{e}') \ast ((v \mapsto \overline{e}') \mathrel{-\!\!*} p)\} \\
\{\exists v'.\ p\} \\
\{p\}.
\end{array}
$$

To go in the other direction, we choose $v'' \notin \mathrm{FV}(e_0, \ldots, e_{n-1}, r)$ to be distinct from v and v', take p to be $\exists v'.\ (v \mapsto \overline{e}') \ast r'$, and use various laws about quantifiers, as well as $r \Rightarrow (q \mathrel{-\!\!*} (q \ast r))$:

$$
\begin{array}{l}
\{r\} \\
\{\forall v''.\ r\} \\
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} ((v'' \mapsto \overline{e}) \ast r)\} \\
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} (((v'' \mapsto \overline{e}') \ast r')/v' \to v)\} \\
\{\forall v''.\ (v'' \mapsto \overline{e}) \mathrel{-\!\!*} (\exists v'.\ (v'' \mapsto \overline{e}') \ast r')\} \\
v := \mathbf{cons}(\overline{e}) \\
\{\exists v'.\ (v \mapsto \overline{e}') \ast r'\}.
\end{array}
$$

Inference Rules for Disposal

The small form (DISSM):

$$\overline{\{e \mapsto -\}\ \mathbf{dispose}\ e\ \{\mathbf{emp}\}}.$$

The large (backward-reasoning) form (DISLG):

$$\overline{\{(e \mapsto -)\ *\ r\}\ \mathbf{dispose}\ e\ \{r\}}.$$

One can derive (DISLG) from (DISSM) by using (FR); one can go in the opposite direction by taking r to be **emp**.

Inference Rules for Lookup

The small form (LKSM):

$$\overline{\{v = v' \wedge (e \mapsto v'')\}\ v := [e]\ \{v = v'' \wedge (e' \mapsto v'')\},}$$

where v, v', and v'' are distinct, and e' denotes $e/v \to v'$.

The large form (LKLG):

$$\overline{\begin{array}{l}\{\exists v''.\ (e \mapsto v'')\ *\ (r/v' \to v)\}\ v := [e] \\ \qquad\qquad\qquad\qquad \{\exists v'.\ (e' \mapsto v)\ *\ (r/v'' \to v)\},\end{array}}$$

where v, v', and v'' are distinct, $v', v'' \notin \mathrm{FV}(e)$, $v \notin \mathrm{FV}(r)$, and e' denotes $e/v \to v'$.

The backward-reasoning form (LKBR):

$$\overline{\{\exists v''.\ (e \hookrightarrow v'') \wedge p''\}\ v := [e]\ \{p\},}$$

where $v'' \notin \mathrm{FV}(e) \cup (\mathrm{FV}(p) - \{v\})$, and p'' denotes $p/v \to v''$.

The large noninterfering form (LKNILG):

$$\overline{\{\exists v''.\ (e \mapsto v'')\ *\ p''\}\ v := [e]\ \{e \mapsto v\ *\ p\},}$$

where $v \notin \mathrm{FV}(e)$, $v'' \notin \mathrm{FV}(e) \cup (\mathrm{FV}(p) - \{v\})$, and p'' denotes $p/v \to v''$.

Inference Rules for Lookup (continued)

To derive (LKLG) from (LKSM), we use the frame rule (FR):

$$\begin{aligned}
&\{v = v' \wedge (e \mapsto v'') * (r/v' \to v)\}\\
&\{v = v' \wedge (e \mapsto v'') * r\}\\
&v := [e]\\
&\{v = v'' \wedge (e' \mapsto v'') * r\}\\
&\{v = v'' \wedge (e' \mapsto v) * (r/v'' \to v)\},
\end{aligned}$$

and the rule for existential quantification (EQ):

$$\begin{aligned}
&\{\exists v''.\ (e \mapsto v'') * (r/v' \to v)\}\\
&\{\exists v', v''.\ v = v' \wedge (e \mapsto v'') * (r/v' \to v)\}\\
&v := [e]\\
&\{\exists v', v''.\ v = v'' \wedge (e' \mapsto v) * (r/v'' \to v)\}\\
&\{\exists v'.\ (e' \mapsto v) * (r/v'' \to v)\}.
\end{aligned}$$

To derive (LKBR) from (LKLG), we take r to be $(e' \mapsto v'') -\!\!* \ p''$ and use the axiom schemata $(e \hookrightarrow e') \wedge p \Rightarrow (e \mapsto e') * ((e \mapsto e') -\!\!* \ p)$ and $q * (q -\!\!* \ p) \Rightarrow p$. (In the first line, we can rename v'' to be any variable not in $\mathrm{FV}(e) \cup (\mathrm{FV}(p) - \{v\})$.)

$$\begin{aligned}
&\{\exists v''.\ (e \hookrightarrow v'') \wedge p''\}\\
&\{\exists v''.\ (e \mapsto v'') * ((e \mapsto v'') -\!\!* \ p'')\}\\
&v := [e]\\
&\{\exists v'.\ (e' \mapsto v) * ((e' \mapsto v) -\!\!* \ p)\}\\
&\{\exists v'.\ p\}\\
&\{p\}.
\end{aligned}$$

Inference Rules for Lookup (continued)

To derive (LKSM) from (LKBR), we take p to be $v = v'' \wedge (e' \mapsto v'')$, after renaming v'' to $\hat{v}$:

$$
\begin{aligned}
&\{v = v' \wedge (e \mapsto v'')\}\\
&\{v = v' \wedge (e \hookrightarrow v'') \wedge (e' \mapsto v'')\}\\
&\{(e \hookrightarrow v'') \wedge (e' \mapsto v'')\}\\
&\{\exists \hat{v}.\ (e \hookrightarrow v'') \wedge \hat{v} = v'' \wedge (e' \mapsto v'')\}\\
&\{\exists \hat{v}.\ (e \hookrightarrow \hat{v}) \wedge \hat{v} = v'' \wedge (e' \mapsto v'')\}\\
&v := [e]\\
&\{v = v'' \wedge (e' \mapsto v'')\}.
\end{aligned}
$$

To derive (LKNILG) from (LKLG), suppose v and v'' are distinct variables, $v \notin \mathrm{FV}(e)$, and $v'' \notin \mathrm{FV}(e) \cup \mathrm{FV}(p)$. We take v' to be a variable distinct from v and v'' that does not occur free in e or p, and r to be $p'' = p/v \to v''$. Then

$$
\begin{aligned}
&\{\exists v''.\ (e \mapsto v'')\ *\ p''\}\\
&\{\exists v''.\ (e \mapsto v'')\ *\ (p''/v' \to v)\}\\
&v := [e]\\
&\{\exists v'.\ (e' \mapsto v)\ *\ (p''/v'' \to v)\}\\
&\{\exists v'.\ (e \mapsto v)\ *\ p)\}\\
&\{(e \mapsto v)\ *\ p)\}.
\end{aligned}
$$

In the first line, we can rename v'' to be any variable not in $\mathrm{FV}(e) \cup (\mathrm{FV}(p) - \{v\})$.

Weakest Preconditions for the New Commands

For each of the new commands, the backward-reasoning form is also a weakest precondition:

Mutation:

$$\overline{[\,(e \mapsto -) \ * \ ((e \mapsto e') \mathrel{-\!\!*} p)\,]_{\mathrm{wp}}\ [e] := e'\ [\,p\,]}.$$

Allocation:

$$\overline{[\,\forall v''.\ (v'' \mapsto e_0, \ldots, e_{n-1}) \mathrel{-\!\!*} p''\,]_{\mathrm{wp}}\ v := \mathbf{cons}(e_0, \ldots, e_{n-1})\ [\,p\,]},$$

where v'' is distinct from v, $v'' \notin \mathrm{FV}(e_0, \ldots, e_{n-1}, p)$, and p'' denotes $p/v \to v''$.

Disposal:

$$\overline{[\,(e \mapsto -) \ * \ r\,]_{\mathrm{wp}}\ \mathbf{dispose}\ e\ [\,r\,]}.$$

Lookup:

$$\overline{[\,\exists v''.\ (e \hookrightarrow v'') \wedge p''\,]_{\mathrm{wp}}\ v := [e]\ [\,p\,]},$$

where $v'' \notin \mathrm{FV}(e) \cup (\mathrm{FV}(p) - \{v\})$, and p'' denotes $p/v \to v''$.

An Annotated Specification

$$
\begin{aligned}
&\{\mathbf{emp}\}\\
&\mathsf{x} := \mathbf{cons}(\mathsf{a}, \mathsf{a}) \;;\\
&\{\mathsf{x} \mapsto \mathsf{a}, \mathsf{a}\}\\
&\mathsf{y} := \mathbf{cons}(\mathsf{b}, \mathsf{b}) \;;\\
&\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{a}) \ast (\mathsf{y} \mapsto \mathsf{b}, \mathsf{b})\}\\
&[\mathsf{x} + 1] := \mathsf{y} - \mathsf{x} \;;\\
&\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x}) \ast (\mathsf{y} \mapsto \mathsf{b}, \mathsf{b})\}\\
&[\mathsf{y} + 1] := \mathsf{x} - \mathsf{y} \;;\\
&\{(\mathsf{x} \mapsto \mathsf{a}, \mathsf{y} - \mathsf{x}) \ast (\mathsf{y} \mapsto \mathsf{b}, \mathsf{x} - \mathsf{y})\}\\
&\{\exists \mathsf{o}.\ (\mathsf{x} \mapsto \mathsf{a}, \mathsf{o}) \ast (\mathsf{x} + \mathsf{o} \mapsto \mathsf{b}, -\mathsf{o})\}
\end{aligned}
$$